Computer Communications 67 (2015) 34–44

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

On service guarantees of fair-queueing schedulers in real systems

Luigi Rizzo [a], Paolo Valente [b,*]

[a] Università di Pisa, Italy
[b] Università di Modena e Reggio Emilia, Italy

ARTICLE INFO

Article history:
Received 18 September 2014
Revised 9 March 2015
Accepted 15 June 2015
Available online 20 June 2015

Keywords:
Packet scheduling
Performance analysis
Service guarantees

ABSTRACT

In most systems, fair-queueing packet schedulers are the algorithms of choice for providing bandwidth and delay guarantees. These guarantees are computed assuming that the scheduler is directly attached to the transmit unit with no interposed buffering, and, for timestamp-based schedulers, that the exact number of bits transmitted is known when timestamps need to be updated.

Unfortunately, both assumptions are unrealistic. In particular, real communication devices normally include FIFO queues (possibly very deep ones) between the scheduler and the transmit unit. And the presence of these queues does invalidate the proofs of the service guarantees of existing timestamp-based fair-queueing schedulers.

In this paper we address these issues with the following two contributions. First, we show how to modify timestamp-based, worst-case optimal and quasi-optimal fair-queueing schedulers so as to comply with the presence of FIFO queues, and with uncertainty on the number of bits transmitted. Second, we provide analytical bounds of the actual guarantees provided, in these real-world conditions, both by modified timestamp-based fair-queueing schedulers and by basic round-robin schedulers. These results should help designers to make informed decisions and sound tradeoffs when building systems.

1. Introduction

Packet schedulers play a critical role in providing bandwidth and delay guarantees on non-overprovisioned transmission links. They are, e.g., one of the key components for guaranteeing the required per-client frame rate and maximum jitter in IPTV managed networks, as well as the required maximum latency in local networks for automotive and avionics applications.

An important family of packet schedulers, namely *fair-queueing* schedulers, originated for the most part as a support for the IntServ QoS architecture [10]. In these schedulers, each packet flow, identified in whatever meaningful way, is associated with a weight, and receives, in the long term, a fraction of the link bandwidth proportional to its weight. Proposed solutions range from plain round-robin [11] to accurate timestamp-based algorithms [1,2,9].

IntServ has basically failed as a QoS architecture in the public Internet. Nevertheless, fair-queueing schedulers or variants of them[1], are now the algorithms of choice in most bandwidth- and delay-sensitive applications, including the previous examples. One of the reasons is that the fair-queueing service scheme easily allows both the desired minimum bandwidth to be guaranteed to each flow or aggregate and the excess bandwidth to be evenly redistributed.

Besides, a series of very efficient yet accurate fair-queueing schedulers has been devised [3,8,13,16]. All these schedulers guarantee a worst-case deviation—with respect to a perfectly fair, ideal service—comparable to that of the optimal WF^{2}Q [2] scheduler. The main practical benefits of such tight service guarantees are a very low jitter and a smooth (non-bursty) service, as thoroughly discussed in [16].

The lowest-cost scheduler in the above series is QFQ+[16], which provides tight guarantees at the amortized cost of just Deficit Round Robin (DRR) [11]. QFQ+ has replaced its predecessor, QFQ [3], in Linux[2] and proved to be even faster than DRR, exactly in the scenarios where using an accurate scheduler matters [16].

Interestingly, the proofs in [16] are based on a slightly more complex system model than that used the in the *classical* analysis of packet schedulers. The reason why a different model has been used coincides with the motivation for this paper: on a real system, packet delays and jitters, as well as per-flow burstiness, may be much higher than predicted by classical analysis. A deeper and general

* Corresponding author. Tel.: +393357182270.
E-mail addresses: rizzo@iet.unipi.it (L. Rizzo), paolo.valente@unimore.it (P. Valente).

[1] Such as *bandwidth servers* in real-time contexts.

[2] QFQ is instead still available in FreeBSD.

http://dx.doi.org/10.1016/j.comcom.2015.06.009
0140-3664/

investigation of this problem was out of the scope of [16], whereas it is exactly the focus of this paper[3].

Problems of classical analysis

Classical analysis is done assuming that the transmit unit is directly attached to the scheduler with no interposed buffering, and, for timestamp-based schedulers, that the exact number of bits transmitted is known on every timestamp update. Neither of these assumptions holds in practice.

First, communication links, especially high-speed ones, are equipped with FIFO queues (sometimes even large ones) to drive the device and absorb the latency and jitter in the hardware and software components that produce packets: memories, buses, interrupt service routines, etc. (in this paper we focus only on FIFO queues in output links). These FIFOs introduce an additional packet delay with any scheduler. Above all, they invalidate, by their very presence, the correctness of fair-queueing timestamp-based schedulers (Section 6.1).

Second, network interfaces do not export a real-time indication of the number of bits transmitted. Deriving this number from the time may be hard too, because, depending on the MAC protocol, the rate may vary with time.

Contributions of this paper

After illustrating the problem with a concrete example, in this paper we make the following contributions:

- We provide a simple and consistent way to modify timestamp-based, worst-case optimal and quasi-optimal fair-queueing schedulers, so as to comply with the presence of FIFOs and with uncertainty on the number of bits transmitted.
- We provide general worst-case bounds on bandwidth, packet (queueing) delay and jitter, for both the resulting family of modified schedulers and basic round-robin schedulers. These bounds take into account exactly the effects of FIFOs and uncertainty on the number of bits transmitted.
- We instantiate and compare these bounds for most schedulers in the above family as well as other popular schedulers.

As for the second contribution, we prove a good and not so obvious result: the worst-case additional packet delay caused by the FIFO, with our FIFO-compliant versions of timestamp-based schedulers, is equal at most to only the time needed to empty the FIFO, although the FIFO not only introduces the latter additional queueing delay, but also perturbs both the packet service order, as we show with a simple example in Section 2, and the timestamp computation (Section 6.1). We also prove an equivalent result in terms of service lag. In other words, we prove that FIFOs cause the minimum possible service degradation.

While partially reassuring, this result means that, in any case, sizing the output queues is critical to avoid that guarantees of sophisticated schedulers degrade to those of, e.g., DRR, or, vice versa, it means that resources should not be wasted on complex scheduling algorithms when short queues are not available. In this respect, our formulas should hopefully help designers find the right compromises between efficiency and guarantees.

Organization of this paper

The rest of this paper is structured as follows. Section 2 shows the problem through a simple example, while Section 3 briefly describes related work. Sections 4 and 5 define the terms used in the paper and provide some background on timestamp-based packet schedulers.

[3] In more detail, in this paper we use the same correct model as in [16], and we report an improved version of the core proofs in [16].

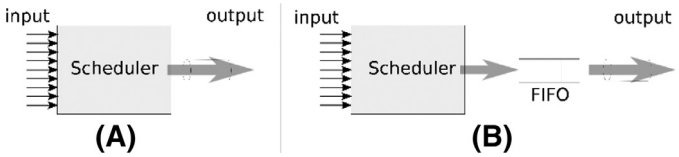

Fig. 1. A: the system model used in the literature, where the scheduler drives directly an ideal link. B: a real system, made of the scheduler followed by an FIFO and the output link.

The core of the paper starts in Section 6, where we define a modification scheme that allows timestamp-based fair-queueing schedulers to comply with the presence of output FIFOs. Section 7 then introduces the service metrics of interest. Resulting service guarantees are computed in Section 9, using the proof machinery provided in Section 8. Finally, Section 10 compares service guarantees with and without queues, and discusses practical implications of our results.

2. A simple example

We start by showing how a queue between the scheduler and the link not only introduces an obvious delay, but can also alter the service order of packets. In the system shown in Fig. 1 on the left, the scheduler is directly connected to the transmit unit, which pushes bits to the communication link as soon as the scheduler makes its decision; this is the idealized model of a system normally considered in the literature. In the system on the right, the scheduler drives instead an FIFO queue which is eventually drained by the transmit unit.

Suppose that in both systems the scheduler is, e.g., WF^{2}Q+ [1], which approximates on a packet-by-packet basis an ideal, infinitely precise subdivision of the link's capacity according to flows' weights (Section 5). Suppose then that at some time t_0 a set of packets arrives simultaneously[4] for flows $P_1 \dots P_N$, all with the same weight $\phi^P = \frac{1}{2N}$. Shortly after t_0, the link becomes ready for transmission, and after another short interval a set of packets arrives for flow Q, which has weight $\phi^Q = \frac{1}{2} \gg \phi^P$. For simplicity assume all packets have length L.

Fig. 2 shows what happens in the system in the two configurations. In the figure, a square represents a packet arriving into the scheduler, a circle indicates a dequeue operation, and a cross indicates the beginning of an actual transmission on the link.

The timing for the idealized case of Fig. 1A is shown in Fig. 2A. Here dequeues and beginnings of transmissions coincide. With our choice of weights, the scheduler correctly services one packet from flows P_i and one from Q, although all the flows P_i were ready before Q.

Fig. 2B shows instead the effect of a FIFO. The FIFO is instantly ready to absorb a large number of packets, so it fills up with most/all packets from flows P_i's before packets from flow Q arrive. Packets from Q therefore find a huge backlog and appear on the link only after this initial burst is complete. As a result, the transmission order and timing looks similar to the one of a DRR [11] scheduler, with or without a FIFO before the link. If the FIFO is large, then the tight service guarantees of WF^{2}Q+ have vanished, and the additional complexity in implementing a scheduler with better guarantees is completely wasted.

In addition, as shown in detail in Section 6.3, a phenomenon like that in Fig. 2B greatly perturbs timestamps in a timestamp-based

[4] Such an arrival pattern is extremely realistic: on many traffic sources or routers, incoming traffic often comes in bursts (corresponding to the processing of a receive interrupt, or the generation of a large TCP segment split into packets). We assume arrivals to be exactly simultaneous for simplicity, but the problem shown in this section can be highlighted also with slightly staggered arrivals, as well as with more complex arrival patterns.